CHAPTER 10

NETWORKING WITH UNIX

After reading this chapter and completing the exercises, you will be able to:

- Describe the origins and history of the UNIX operating system
- Identify similarities and differences between popular implementations of UNIX
- Understand why you might choose a UNIX server for a corporate network
- Explain and execute basic UNIX commands
- Install Linux on an Intel-based PC
- Use Linux to add groups and users and to change file access permissions
- Explain how UNIX can be internetworked with other network operating systems

ON THE JOB

I started working with UNIX systems in the early 1980s. I learned UNIX (and the C programming language) at the same time I learned CP/M and MS-DOS, but I tended to prefer the UNIX systems. The most attractive part of UNIX was the incredible power of the command interpreter (or shell). With the UNIX shell, I could create complex new commands out of a few existing commands. I could accomplish things that the programmers of the original commands never imagined. The community of UNIX programmers enjoys thinking about and tinkering with the system, aiming to continually improve it.

The other significant facet of the UNIX community is the support that people offer to each other. It's nearly guaranteed that someone else has experienced the problem you're facing at any given time. A simple query posted to the Internet often receives an answer within a few hours. The low number of UNIX-based computer viruses illustrates this community's supportive attitude.

Networking with UNIX is simple and straightforward. The UNIX shell is a great place to experiment with networking ideas because it's easy to test those ideas by typing simple commands in the shell and watching the results. With the advent of Linux and other open UNIX-like operating systems, the opportunity to tinker with networking is now readily available to many more people. It's easy for me to understand why UNIX in general and Linux in particular have experienced such explosive growth in the past few years.

David Klann
Berbee Information Networks, Inc.

Along with Windows 2000 and Novell NetWare, UNIX is one of the most popular network operating systems. Although all three operating systems enable servers to provide resource sharing, UNIX differs in fundamental ways from NetWare and Windows 2000. Researchers at AT&T Bell Laboratories developed UNIX in 1969; thus it is much older than NetWare and Windows 2000. In fact, UNIX preceded and led to the development of the TCP/IP protocol suite in the early 1970s. For this reason, it is considered the parent of TCP/IP networking. Today, most Internet servers are computers running UNIX. Reflecting this operating system's efficiency and flexibility, the number of UNIX systems continues to grow.

Many local and wide area networks include servers that run UNIX. You should be familiar with UNIX so you can set up and maintain these networks. Mastering UNIX can be complicated because it is not controlled and distributed by a single software manufacturer. Instead, numerous vendors sell a number of varieties, such as Solaris from Sun Microsystems, AIX from IBM, and HP-UX from Hewlett-Packard. Each of these varieties is known as a proprietary implementation, or, more casually, as a "flavor." Although the various flavors of UNIX follow some accepted standards, each has unique characteristics. In addition, nonproprietary, freely distributed UNIX-like implementations such as Linux, the GNU Hurd, and FreeBSD are available. Fortunately, the differences between implementations are relatively minor, and with a little effort you can understand them and move from one implementation to another with ease. This chapter introduces the UNIX operating system in general and describes Linux in more detail.

> Note
>
> UNIX and Linux share many characteristics, but are subtly and often confusingly different. UNIX is the trademarked name given to the operating system originally developed at Bell Labs. A nonprofit industry association named The Open Group owns the trademark. An operating system must pass The Open Group's qualification tests to be called UNIX. Linux grew out of an independent effort to create an operating system that behaves like the trademarked UNIX operating system. This would be equivalent, in the Windows world, to a large group of people getting together and writing a version of Windows 2000 based on the public specifications. (UNIX, however, has been publicly available for much longer than Windows 2000, and Microsoft keeps the specifications to many parts of Windows 2000 a closely guarded secret.)

A Brief History of UNIX

The UNIX operating system is characterized by a rich tradition and a culture of personal friendships in an era of impersonal disconnectedness that, some would say, the computer itself has fostered.

In the late 1960s, a few programmers grew dissatisfied with the existing programming environments. In particular, they didn't like the cumbersome nature of the existing systems that required a programmer to write a set of instructions, submit them all at once, and then wait for the results. Instead, programmers desired a more interactive operating environment that allowed them to build and test their programs piece by piece. In addition, the programmers sought a system that imposed as few predetermined structures as possible on the users. Structures that they chose to leave up to users and application programmers include the format of data within data files and the notion of assigning significance to filenames. For example Windows assigns meaning to the period (".") and the last three characters of filenames. UNIX imposes no file-naming constraints. Two employees at Bell Labs in Murray Hill, New Jersey, Ken Thompson and Dennis Ritchie, decided to overcome the limitations of existing operating systems by creating an entirely

new programming environment. To properly design this new environment, they decided to start at the lowest level—at the operating system. This environment ultimately evolved into the UNIX operating system.

Antitrust law prohibited AT&T from profiting from the sale of computers and software during the 1970s. Thus, for a nominal licensing fee, anyone could purchase the source code to the work produced at Bell Labs. The word spread rapidly, and researchers in educational institutions and large corporations all over the world soon had this curious new software running on their lab computers. Versions of UNIX that come from the Bell Labs are known as **System V**. Researchers at the University of California at Berkeley were among the first enthusiastic supporters of early versions of UNIX. They added many useful features to the system, including the TCP/IP network subsystem. Berkeley versions of UNIX are known as **BSD (Berkeley Software Distribution)**.

The 1980s saw the breakup of AT&T. This event enabled the company to begin actively marketing the UNIX system to other computer manufacturers. After a number of fits and starts, AT&T eventually sold its rights to the UNIX system. These rights changed hands a number of times during the early 1990s. Today, ownership of the UNIX system is shared by three organizations—Caldera International, Inc., Tarantella, Inc., and The Open Group.

Caldera International and Tarantella (formerly The Santa Cruz Operation), jointly own the rights to the UNIX source code. They therefore have the right to distribute copies of the **source code**—the raw materials for creating a UNIX system. Anyone could write a UNIX operating system, but the effort required to do so is prohibitive. Most organizations choose to start with the existing source code by obtaining it from Caldera International and Tarantella and then make modifications for their specific computer hardware.

The Open Group, as mentioned earlier, owns the UNIX trademark. After a vendor changes the code licensed from Caldera International and Tarantella, its modified system must pass The Open Group's verification tests before the operating system may be called UNIX. Compaq, for example, pays source code licensing fees to Caldera International or Tarantella and verification and trademark use fees to The Open Group so that it can call its operating system Tru64 UNIX.

Although many versions of UNIX may be used as network operating systems, all UNIX versions share the following features:

- The ability to support multiple, simultaneously logged in users
- Hierarchical file systems that incorporate demountable volumes
- Consistent interfaces for input of data to and output of data from hardware devices, files, and running programs
- The ability to start processes in the background
- Hundreds of subsystems, including dozens of programming languages

- Program source code portability
- Window interfaces that the user can configure, the most popular of which is the X Window system

You will learn more about the UNIX memory model, file system, processing capabilities, and network integration later in this chapter.

THE CURRENT STATE OF THE MARKET

The UNIX market is huge and highly segmented. As a result, stating its size with precision is difficult. People use UNIX-based systems for everything from running general-purpose workstations to controlling industrial robots and telecommunications equipment. In fact, even some **real-time** implementations of the UNIX system exist, in which the operating system must respond to input immediately. One such implementation, QNX, is used to run a computer vision system for the NASA space shuttle and the international space station.

Proprietary implementations and open source implementations are two of the most significant UNIX market segments, as explained in the following sections.

Proprietary UNIX

Many companies market both hardware and software based on the UNIX operating system. An implementation of UNIX for which the source code is either unavailable or available only by purchasing a licensed copy from Caldera International and Tarantella (costing as much as millions of dollars) is known as **proprietary UNIX**. By most counts, the three most popular vendors of proprietary UNIX are Sun Microsystems, IBM, and Hewlett-Packard. Sun's proprietary version of UNIX, called **Solaris**, runs on the company's proprietary SPARC-based workstations and servers, as well as Intel-based Pentium-class workstations and servers. IBM's proprietary version, **AIX**, runs on its PowerPC-based RS-6000 computers. HP's proprietary version, **HP-UX**, runs on its PA-RISC–based systems. Many other organizations have licensed the UNIX source code and created proprietary UNIX versions that run on highly customized computers (that is, computers that are appropriate for very specific tasks).

Choosing a proprietary UNIX system has several advantages:

- *Accountability and support*—An organization might choose a proprietary UNIX system so that when something doesn't work as expected, it has a resource on which to call for assistance.
- *Optimization of hardware and software*—Workstation vendors who ship proprietary UNIX invest a great deal of time in ensuring that their software runs as well and as fast as possible on their hardware.

- *Predictability and compatibility*—Purveyors of proprietary UNIX systems strive to maintain backward-compatibility with new releases. They schedule new releases at somewhat regular, predictable intervals. Customers usually know when and how things will change with proprietary UNIX systems.

One drawback of choosing a proprietary UNIX system, however, relates to the fact that the customer has no access to the system's source code and, thus, cannot customize the operating system. Open source UNIX solves this problem.

Open Source UNIX

An interesting factor in the UNIX marketplace over the past few years has been the emergence of UNIX-like systems that are not owned by any one company. This software is developed and packaged by a few individuals and made available to anyone, without licensing fees. Often referred to as **open source software**, or **freely distributable software**, this category includes UNIX-like systems such as the **GNU** (whose kernal is known as Hurd), **FreeBSD**, and **Linux**. Each of these systems, in turn, comes in a variety of implementations with slightly different features and capabilities. As mentioned, these packages are often referred to as the different **flavors** of the open source software. For example, the different flavors of Linux include RedHat, Caldera, Mandrake, and a host of others. While these packages are free of licensing fees and are available at no cost to users, you can purchase the software. In the case of purchasing Red Hat Linux, for example, you are paying for the convenience of a package that includes the software on CD-ROM, documentation, and access to Red Hat's customer support. Choosing to obtain Red Hat Linux for free means a (potentially) long download (more than 650 megabytes), writing the software to a CD-ROM, and learning how to install it without the aid of printed documentation. Many people find the convenience worth the nominal purchase price of software with a licensing fee.

The key difference between freely distributable UNIX and proprietary implementations of UNIX relates to the copyright specification. Freely distributable versions of UNIX include a copyright (called the **General Public License**) that *requires* the source code to be made available to anyone receiving the system. What's more, this license requires programmers to publish source code changes so that others can make use of them. This ensures that programmers have access to the original source code and each change made to it. The General Public License further ensures that no one person or entity can claim ownership of the source code. Although Linus Torvalds holds the copyright to the Linux operating system, even he cannot prevent others from obtaining and sharing the source code. A primary advantage of open source UNIX is that users can add functionality not provided by a vendor of proprietary UNIX. A manufacturing company that uses computer-controlled robotic spot welders, for example, might combine open source UNIX with custom software to control its robots. In contrast, it might be very difficult or costly to integrate the robotic control software with a proprietary UNIX system.

Versions of freely distributable UNIX run not only on Intel-based processors, but also on other processor brands such as PowerPC (used in Apple Macintoshes), SPARC (used in Sun Microsystems workstations), and Alpha (used in Compaq workstations). Although this range of choices is wider than with proprietary UNIX systems, it can also complicate the decision-making process when choosing a network server.

The discussion of UNIX in the remainder of this chapter will focus on one popular open source version, Linux. Linux follows standard UNIX conventions, is highly stable, and is free. Linus Torvalds, then a second-year Finnish computer science student, developed it in 1991. After developing Linux, Torvalds posted it on the Internet and recruited a number of other UNIX aficionados and programmers to help enhance it. Today, Linux is used for file, print, and Web servers across the globe. Its popularity has even convinced large corporations that own proprietary UNIX versions, such as IBM, Silicon Graphics and Sun Microsystems to publicly embrace and support Linux.

WHY CHOOSE UNIX?

Let's say that your supervisor assigns you the task of choosing and installing a new server on your organization's LAN. To make your network as compatible as possible with other networks, you limit your operating system options to the big three: NetWare, Windows 2000, and UNIX. Given these options, what considerations might lead you to choose UNIX? The same set of questions asked in Chapters 8 and 9 apply to your consideration of this operating system:

- Can it be integrated with my existing infrastructure?
- Will it provide the security required by my resources?
- Is my technical staff capable of managing it?
- Will my applications run smoothly on it?
- Will it accommodate future growth (that is, is it scalable)?
- Does it support the additional services required by my users (for example, remote access, Web site hosting, and messaging)?
- How much does it cost?
- What kind of support does the vendor offer?

UNIX systems offer a host of features, including the TCP/IP protocol suite and all applications necessary to support the networking infrastructure as a part of the basic operating system. You get the programs necessary to perform operations such as routing, firewalling, domain name service, and automatic IP address assignment when you install the system on your computer. UNIX supports non-IP protocols such as Novell's IPX/SPX and AppleTalk. Like Windows 2000 and NetWare, UNIX also supports many different network topologies and physical media, including Ethernet, Token Ring, FDDI, and wireless LANs.

UNIX systems can act as file servers to Windows, NetWare, and Macintosh clients. The open source software package called **Samba**, for example, is a complete Windows 2000–style file and printer sharing facility. Other proprietary and open source software packages are available that implement the NetWare file and print server facilities as well as Macintosh file and print facilities.

UNIX efficiently and securely handles the growth, change, and stability requirements of today's diverse networks. The source code on which UNIX systems are based is mature, as it has been used and thoroughly debugged for nearly 30 years. Like NetWare, UNIX allows you to change the server's configuration—for example, assigning a different IP address to an interface—without restarting the server. Similarly, you can easily modify a UNIX system while it is running. When you need to access a tape drive, for example, you can enable the tape driver, access the tape, and disable the tape driver without restarting the server. This functionality allows you to use memory on your server very efficiently.

A key difference between UNIX and other network operating systems such as NetWare relates to resource sharing. UNIX was originally developed as a **time-sharing system**—that is, a computing system to which each user must attach directly (usually with a "dumb terminal") to share the resources of that computer. You must log onto a UNIX system and run applications on the system to share its resources.

To understand how this model differs from other client/server resource-sharing methods, consider your company's office environment. You share certain resources: file cabinet space, conference rooms, printers, copiers, and so on. When you are working in your office, you share office resources in the same way that a UNIX system shares its resources. For example, you might walk over to the file cabinet, remove a file, and then replace it when you're done with it.

In the NetWare model, in contrast, a workstation is attached to the network. When you want to use the server's resources, you simply map a drive to your local computer or send a print job to the server's printer queue. This approach is analogous to a telecommuting situation in which you work at home, but still have access to many of the office's resources. To share your office resources, you can simply call someone at the office and ask to have a paper file sent to your home by courier. Later, the courier can shuttle the file back to the office and replace it in the file cabinet. The disadvantage of this model is that you cannot use some of the office's resources, such as the conference rooms, when you telecommute. Similarly, you can't perform some tasks on a NetWare server because you don't attach directly to the system; instead, you simply use its resources over the LAN.

A real advantage of UNIX is that people have added applications and services that enable sharing of resources in the telecommuter model without eliminating the "drive-to-the-office" model. With UNIX systems, you get the best of both worlds.

UNIX systems also include a robust and mature security model. Some proprietary UNIX systems have even received **Orange Book** certification, which is a rigorous operating system security specification that the U.S. Department of Defense first published in 1985. These characteristics of the UNIX system contribute to its ability to handle networks' growth and changes, and provide stability.

UNIX Server Hardware

UNIX systems may be implemented as workstations or as servers. Unlike Windows 2000, in which the server and workstation versions vary considerably, a UNIX server and a UNIX workstation differ only by the set of optional packages included during installation. A UNIX system configured as a server has the necessary software to enable sharing of resources such as print queues, file systems, and processor time. Hardware requirements are very similar to those for NetWare and Windows 2000 servers.

In UNIX, the use of a graphical user interface (GUI) remains optional—that is, you choose to use the GUI, a command-line interface, or a combination of the two. By contrast, in all versions of NetWare except 5.0 and later, you must use a character-based console (one which uses textual menus and submenus). In Windows 2000, on the other hand, you *must* use the GUI for many operations. Many people see the flexibility of UNIX as an advantage. For example, you might choose to use the GUI for operations that require a great deal of user interaction such as adding new users or configuring services. However, for server operations that run unattended, it often makes sense to use the command-line interface (which uses up less of the computer's memory and other resources).

The hardware for a UNIX server consists of a base system unit, which must include the following equipment:

- A motherboard with CPU, memory, and I/O control
- A network interface card (NIC)
- A floppy disk drive
- A CD-ROM drive
- One or more fixed disks

High-performance video cards, sound cards, and other I/O devices are optional. Your major decisions in choosing the hardware for a general-purpose UNIX server can be summarized as follows:

- Which applications and services will run on the server?
- How many users will this system serve?
- How much random access memory (RAM) will the server need?
- How much secondary storage (hard disk) will the server need?

Table 10-1 shows the minimum hardware requirements for the various components of a Linux server. You may find more current lists of supported hardware on the hardware compatibility list (HCL) at *www.linuxdoc.org/HOWTO/Hardware-HOWTO/*.

Table 10-1 Typical hardware requirements for a Linux server

Component	Requirement	Notes
Processor	Intel-compatible x86	Recent versions of Linux (2.0 and later) include support for as many as 16 Intel processors.
Memory	32 MB RAM	You should consider more RAM than this minimum for better performance; most network administrators opt for 64 MB or 128 MB of RAM for servers.
Hard disk	A hard drive supported by Linux (as specified in the HCL), such as an IDE or SCSI, with a minimum of 500 MB free space	Most server implementations require additional hard drive space.
NIC	A NIC supported by Linux (as specified in the HCL)	
CD-ROM	Choose a drive listed on the HCL.	Recent versions of Linux support SCSI, IDE, and ATAPI CD-ROM drives.
Floppy disk	Without a bootable CD-ROM drive, Linux installation (and most other UNIX systems) requires one or two 3.5-inch floppy disks to get started. You may create emergency repair disks during installation.	
Pointing device	Optional	Only necessary if you install the GUI component.

The Linux HCL resembles the NetWare and Windows 2000 hardware guides in that it recommends *minimum* requirements for simply starting the server. You'll need to add more memory and more disk space according to your applications' requirements. Unfortunately, you sometimes cannot learn the memory requirements of an application until you actually run it on the server. In these instances, it is always better to overestimate your needs than to underestimate them.

No one "right" UNIX server configuration exists. To determine the hardware requirements of your UNIX server, you need to take into account the nature of the work to be performed by the system, the type of software to be run, and the number of users.

A Closer Look at Linux

Linux is the third major network operating system discussed in this book. As you probably realize, both similarities and differences exist among Linux, NetWare, and Windows 2000. This section compares Linux with those other network operating systems.

Linux Multiprocessing

Any modern network operating system must use the resources of multiple processors in an efficient manner. Linux is truly a modern operating system in this respect. Like NetWare and Windows 2000, Linux supports symmetric multiprocessing (SMP). This support for SMP was experimental in version 2.0. However, Linux experts consider SMP to be stable in versions 2.2 and later. SMP support and performance improved with the release of Linux version 2.4. The operating system supports SMP using a maximum of 16 processors per server. The guidelines outlined in the previous chapters for NetWare and Windows 2000 apply to Linux as well. You must know how your servers will be used and plan for multiprocessing servers according to your estimated application processing loads.

Default Linux installations prior to version 2.2 leave SMP disabled. You must configure your system so as to enable SMP if you're using version 2.0. Consult the Linux SMP guide at *www.linux.org.uk/SMP/title.html* for details.

The Linux Memory Model

From its inception, Linux was created to use both physical and virtual memory efficiently. (See the "Windows 2000 Server's Memory Model" section in Chapter 8 for details.) Like Windows 2000, Linux allocates a memory area for each application. It attempts to decrease the inefficiency of this practice, however, by sharing memory between programs wherever it can. For example, if five people are using FTP on your Linux server, five instances of the FTP program will run. In reality, only a small part of each FTP program (called the private data region—the part that stores the user name, for example) will receive its own memory space; most of the program will remain in a region of memory shared by all five instances of the program. In this case, rather than using five times the memory required by one instance of the program, Linux sets aside only a little more memory for five FTP users than it does for one FTP user.

Most current versions of Linux use a 32-bit addressing scheme that enables programs to access 4 GB of memory. Linux also runs on CPUs that employ 64-bit addresses, enabling programs to access more than 18 exabytes (2^{64} bytes) of memory. That's more than 18 billion billion bytes of data; three times the total number of words ever spoken by human beings, by one estimate! Virtual memory in a Linux server can take the form of a disk partition (created with the Windows fdisk command), or it can be in a file (much like the virtual memory file pagefile.sys in Windows 2000).

The Linux Kernel

Linux is similar to NetWare in that the core of the system consists of the **kernel**. The Linux kernel is loaded into memory from disk and runs when you turn on your computer. Also, as with NetWare, you can add or remove functionality by loading and unloading Linux **kernel modules**, which are analogous to NetWare NLMs. Unlike NetWare, however, Linux does not use loadable modules exclusively to extend the functionality of the system. Rather, you start and stop Linux services and applications by typing commands

(much like running commands in a Windows 2000 command window). Linux offers the best of both worlds: loadable kernel modules to extend the functionality of the Linux kernel (like NetWare), and services and applications that run to perform most of the work of the server (like Windows 2000).

Note

When vendors speak of Linux version numbers, such as 2.2 or 2.4, they are referring to the kernel version. Although vendors (such as Red Hat) may also use their own version-numbering scheme (such as Red Hat 7.0), all purveyors of Linux distributions use the same Linux kernel version scheme to identify which kernel is included in the package.

Linux File and Directory Structure

The UNIX system was one of the first operating systems to implement a **hierarchical file system**. This hierarchy somewhat resembles the structure of the NDS inverted tree discussed in Chapter 9. The notion of a file system organized in this way was considered revolutionary at the time of UNIX's inception. Today, most operating systems, including all Microsoft operating systems, NetWare, and even the Apple Macintosh's MacOS, use hierarchical file systems. Figure 10-1 shows a typical UNIX and Linux file system hierarchy.

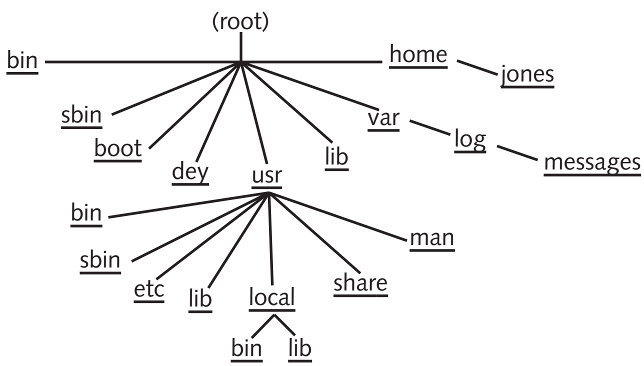

Figure 10-1 Linux file system hierarchy

The */boot* directory contains the Linux kernel and other system initialization files. Linux keeps applications and services in */bin and /sbin* (applications and services in */sbin* support the system initialization process; you'll rarely use these programs). The */var* directory holds variable data (such as log files and print jobs waiting to be printed). The file */var/log/messages*, for example, stores system log messages. Users' login directories appear in */home*. When you create a new user account, the system assigns a directory in */home* to that user. The login (or home) directory matches the account's user name. Thus */home/jones* is the login (or home) directory for the user *jones*.

Linux File Services

In Chapter 8, you learned about the file systems supported by Windows 2000. As you will recall, the file system constitutes the operating system's method for organizing, managing, and accessing its files through logical structures and software routines. Linux includes support for multiple types of file systems, including local and remote file systems. The native file system type, called *ext2*, is the "second extended" file system for Linux. Rather than trying to fix some of the problems with the "first extended" file system, the programmers decided to create a new file system from scratch, and named it (creatively) the second extended file system.

Linux allows you to access partitions formatted with the DOS FAT file system as well as the Windows 2000 NTFS file system (in read-only mode) and the OS/2 HPFS file system. It also supports remote file systems, which are analogous to Windows shares or NetWare network volumes. With Linux, you can both map shared file systems (drives) from Windows or NetWare servers and share local partitions with other users. Sun Microsystems' **Network File System (NFS)** is the other significant remote file system type supported by Linux. Given how many file system types are accommodated, it's easy to see why Linux has become such a popular server platform in large, diverse networks.

Linux Internet Services

As you learned earlier, UNIX has deep roots in Internet services. For instance, the leading Internet Web server is an open source software application called **Apache** that originally ran only on UNIX systems. What's more, the original Web tools—including the first browsers and servers—were developed on UNIX-based systems. UNIX-based systems acted as the development platforms for the original ARPAnet and Internet services such as FTP, Telnet, gopher, HTTP, and POP. It should, therefore, come as no great surprise that current implementations of UNIX include the full range of Internet services as standard components.

Linux Processes

Another UNIX innovation is the notion of separate, numbered processes. Each process represents an instance of a running program in memory (RAM). The UNIX kernel allocates separate resources (such as memory space) to each process as it is created. It also manages all programs' access to these resources. This novel approach enables partitioning of processes in memory, thereby preventing one program from disrupting the operation of the entire system. When one program ends unexpectedly on a UNIX system, it doesn't cause the whole computer to crash. In addition to processes, Linux also supports threads, discussed in Chapter 8.

A Linux Command Sampler

The command line is the primary method of interacting with a Linux system. Even when you're running a GUI, the GUI actually executes commands on your behalf in response to your manipulation of the graphical elements on the screen. This section discusses some of the basics of the Linux user interface, interaction with the Linux command line, and some fundamental Linux commands.

The program that accepts your typing and runs the commands for you is called a **command interpreter**. Also known as a **shell**, a command interpreter translates your typed commands into machine instructions that the operating system can understand. Thus, the command interpreter is a program that runs other programs. UNIX command interpreters also perform file globbing (described later) and keep track of the command history (much like the **doskey** command in DOS and Windows 2000). The primary UNIX command interpreter is */bin/sh*. It has a similar function to the primary Windows 2000 command interpreter, *cmd.exe*. To use the shell effectively, you should be familiar with at least some basic commands.

One especially useful feature of UNIX (including Linux) is its online documentation. UNIX systems include documentation, known as the **manual pages**, for all commands. You can review the instructions for any command by reading its manual page entry. Although their organization differs slightly in various flavors of UNIX, manual pages are typically arranged in nine sections:

- *Section 1* covers the commands that you most typically enter while typing in a command window.
- *Sections 2 through 5* document the programmer's interface to the UNIX system.
- *Section 6* documents some of the amusements and games that are included in the UNIX system.
- *Section 7* describes the device drivers for the system.
- *Section 8* covers the commands used by administrators to manage the system.
- *Section 9* documents the UNIX kernel functions programmers use when writing device drivers.

You can access manual pages by entering the `man` command in a Linux command window. For example, to read the manual page entry for the `telnet` command, you would type `man telnet` in a command window, and then press Enter.

Although the Linux manual pages are accurate and complete, Linux newcomers often complain that they can't find the appropriate manual page if they don't know the name of the command they want to use. That's why the `apropos` command exists. It enables you to find possible manual page entries for the command you want to use. For example, you might type `apropos list` to search for a command that lists files. The `apropos` command would then display all commands and programming functions that include the

keyword *list* in their manual page entries. Type `man <command>` (where *<command>* is a command name displayed by apropos) when you find a command name that looks like it might do what you want.

Commands function in much the same way as sentences in ordinary language. Some sentences are one-word directives to the system requesting that it perform a simple task on your behalf (such as *date* for "tell me the current date and time"). Other sentences are detailed instructions to the system containing the equivalent of nouns, adjectives, and adverbs and creating a precise description of the task you wish the system to perform. For example, to instruct the system to "print the names of all files in the current directory that have been accessed in the past five days," you would type: `find. –type f –atime +5 –print`.

A few rules exist to guide your use of Linux commands and, as you might expect, exceptions to most of the rules also exist. Most commands (though not all) are lowercase alphabetic characters. Using the analogy of a sentence, the command itself would be the verb—that is, the action you want the system to take (for example, *ls* to list information about files). The things on which you want the system to operate (often files) would be the nouns. (So for example, you would type `ls accounts.xls` to list a file named account.xls.) Options to the commands are analogous to adjectives and the adverbs—that is, modifiers that give more specifics about the command. To specify an option, you usually type a dash (-) followed by a letter (such as *ls -a* to list all files, even the "invisible" files in the current directory). You can make commands even more specific by using file **globbing**—the equivalent to using wildcards in Windows and DOS. On a Linux system, this operation is also called filename substitution (for example, `ls –l a*` would produce a detailed listing of all files beginning with the letter "a").

A significant (and perhaps initially confusing) difference between the Linux and Windows 2000 command-line interfaces relates to the character you use to separate directory names when you type in a command window. The Windows 2000 separator character is "\" (backslash). The equivalent Linux directory separator character is "/" (forward slash). For example, in a Windows 2000 command window, you type the *telnet* command as `\windows\system32\telnet.exe`. The `telnet` command in Linux is `/usr/bin/telnet`.

Windows 2000 and UNIX share the powerful concept of directing output from one command to the input of another command. A **pipe** (entered as a vertical bar "|") serves as the connection between two commands. Think of data "flowing" from one command to another. Most commands that display output on the monitor allow you to direct the output to another command. Most commands that accept typing from your keyboard also accept input from other commands. Two or more commands connected by a pipe are called a **pipeline**. UNIX pipes make it easy to create sequences of commands that might require custom programming on other systems.

Table 10-2 lists some common Linux commands and provides a brief description of each.

> **Note**
>
> The developers of the original UNIX system worked at AT&T, then the largest public corporation in the world. Two features of communication within large corporations are the tendency to abbreviate words and the reliance on acronyms. The command names in the Linux system reflect this culture in that they drop vowels and syllables (*cp* for *copy*, *cat* for *concatenate*, and so on), and name commands with the "initials" of their intended use (`grep` for *general regular expression parser*, `ftp` for *File Transfer Protocol*). Refer to the relevant manual pages when you encounter command names that you don't understand. The *synopsis* section usually indicates the origin of the command name.

Table 10-2 Commonly used Linux commands

Command	Function
date	Display the current date and time.
ls –la	Display with details all the files in the current directory.
ps –ef	Display details of the current running programs.
find *dir* *filename* –print	Search for *filename* in the directory *dir* and display the path to the name on finding the file.
cat *file*	Display the contents of *file*.
cd */d1/d2/d3*	Change the current directory to *d3*, located in */d1/d2*.
cp *file1* *file2*	Make a copy of *file1*, named *file2*.
rm *file*	Remove (delete) *file* (Note that this is a permanent deletion, there is no trash can or recycle bin from which to recover the deleted file.)
mv *file1* *file2*	Move (or rename) *file1* to *file2*.
mkdir *dir*	Make a new directory named *dir*.
rmdir *dir*	Remove the directory named *dir*.
who	Display a list of users currently logged in.
vi file	Use the "visual" editor named *vi* to edit *file*.
grep *"string"* *file*	Search for the string of characters in *string* in the file named *file*.
ifconfig	Display the network interface configuration, including the IP address, MAC address, and usage statistics for all network interface cards in the system.
netstat –r	Display the system's TCP/IP network routing table.
sort *filename*	Sort alphabetically the contents of *filename*.
man *"command"*	Display the manual page entry for "*command*."
chmod *rights* *file*	Change the access rights (the mode) of *file* to *rights*.
telnet *host*	Start a virtual terminal connection to *host* (where *host* may be an IP address or a host name).
ftp *host*	Start an interactive file transfer to (or from) *host* using the FTP protocol (where *host* may be an IP address or a host name).

Table 10-2 Commonly used Linux commands (continued)

Command	Function
`startx`	Start the X Window system.
`kill` *`process`*	Attempt to stop a running program with the process ID *process*.
`tail` *`file`*	Display the last 10 lines of *file*.
`exit`	Stop the current running command interpreter. Log off the system if this is the initial command interpreter started on logging in.

The most frequently used Linux command is *`ls`*. By entering *`ls`* (and specifying *`-l`*, the detailed listing option), you learn everything about a file except its contents. Linux systems keep quite a bit of information about each file including:

- The filename
- The file size (in bytes)
- The date and time that the file's i-node (file information node, discussed below) was created
- The date and time that the file was last accessed (viewed or printed)
- The date and time that the file contents were last modified (created, edited, or changed in any way)
- The number of "aliases" or links to the file
- The numeric identifier of the user who owns the file
- The numeric identifier of the group to which the file belongs
- The access rights for the owner, the group, and all others

The system stores this information for each file (except the filename) in a file information node (abbreviated **i-node)**. The beginning of each disk partition contains reserved space for all i-nodes on that partition. I-nodes also contain pointers to the actual file contents on the disk. The file's name is stored in the directory that contains the file. To learn about the i-node information, you use the `ls` command. Figure 10-2 shows a sample list generated by `ls`.

```
% ls -l
total 1278
drwxr-xr-x   2 root     bin          2048 Dec 11 17:05 bin
drwxr-xr-x   2 root     root         1024 Sep 24 01:38 boot
drwxr-xr-x   2 root     root         1024 Feb 26  1998 cdrom
drwxr-xr-x   3 root     root        19456 Jan 16 02:52 dev
drwxr-xr-x   9 root     root         3072 Feb  2 07:27 etc
drwxr-xr-x   2 root     root         1024 May 27  1997 floppy
drwxr-xr-x  78 root     root         2048 Jan 30 11:55 home
drwxrwxr-x   4 root     root         1024 Dec 15 13:12 home2
drwxr-xr-x   3 root     root         1024 Sep 28 13:18 lib
drwxr-xr-x   2 root     root        12288 May 27  1997 lost+found
drwxr-xr-x   2 root     root         1024 Jun 21  1996 mnt
drwxrwxr-x   9 root     root         1024 Jan 16 05:42 nfs
dr-xr-xr-x   5 root     root            0 Jan 15 20:52 proc
drwxr-x--x   6 root     root         1024 Dec 22 05:56 root
drwxr-xr-x   2 root     bin          2048 Jul  7  1998 sbin
drwxr-xr-x   2 root     root         1024 May 27  1997 shlib
drwxrwxrwt  45 root     root        12288 Feb  2 07:56 tmp
drwxrwxr-x  24 root     root         1024 Jan 16 03:54 usr
drwxr-xr-x  17 root     root         1024 Dec 14 08:10 var
-r--------   1 root     root       359327 May 27  1997 vmlinuz-2.0.29
-r--------   1 root     root       468684 May 27  1997 vmlinuz-2.0.29-1
-r--------   1 root     root       404117 Apr  8  1998 vmlinuz-2.0.33
drwxrwxr-x   5 root     binc         1024 Jan 16 05:56 www
%
```

Figure 10-2 Example of output from `ls`

The strings of r's, w's, x's, and so on, in the left column, represent the access permissions for the files. The first character in the access permissions field (on the far left) indicates the file type. Files with a type of "d" are directories. Files whose type is shown with a "-" are regular files such as word-processing files or spread sheet files—that is, they simply contain unstructured (to the operating system) data. Other valid file types are as follows:

- "l" for symbolic link files (much like Windows 2000 shortcuts)
- "b" for block device files (such as disk partitions)
- "c" for character device files (such as serial ports)

Figure 10-3 shows how to interpret the rest of the output of `ls`.

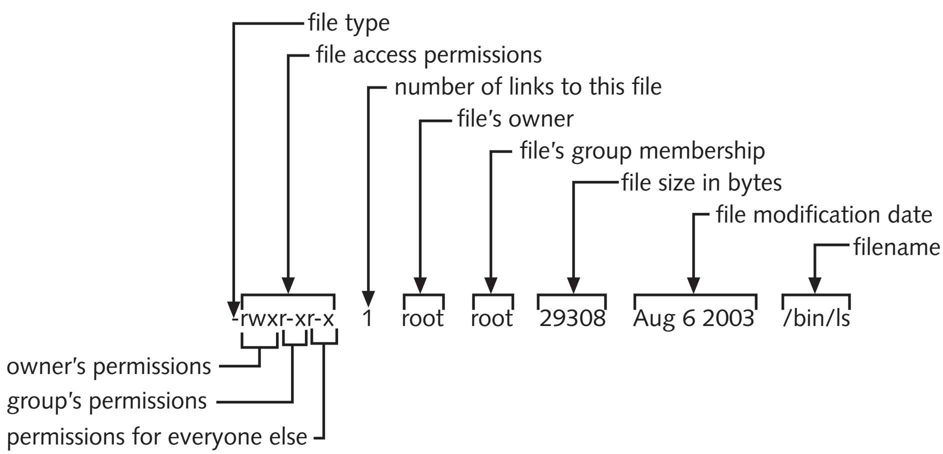

Figure 10-3 Anatomy of `ls` output

INSTALLING AND CONFIGURING A LINUX SERVER

You've had a taste of the Linux system. Now it's time to build one. This section will walk you through the installation process for the Intel-based Red Hat Linux. Although you can install Red Hat Linux over a network using FTP or NFS, this section describes the quickest way—using a CD-ROM.

Preinstallation Requirements

In previous chapters, you learned about the importance of thorough planning in the installation of a new server. These considerations apply to Linux as well as to Windows 2000 and NetWare. Although making changes to the server setup after you install a Linux system is easier and less disruptive than with Windows 2000, you should nevertheless plan as carefully as possible to avoid service unavailability after a Linux server is running.

Before installing Linux, you should be prepared to answer the following questions:

- *What is the new server's name?* This name is a less important issue for Linux systems than for Windows 2000 or NetWare systems, but it's still a good idea to choose it before beginning installation. You can add the server's name to your network name service (DNS, for example) as soon as you choose it. (See Chapter 11 for details about TCP/IP domain names.) You may use any name containing a maximum of 32 alphanumeric characters, not including the following:

  >< [ ]._:; | = , + * " ?

- *What is the server's IP address?* You'll need this address to enable the network on the new server. Network administrators usually configure workstations to obtain an IP address automatically when they start up, but they often configure servers with reserved or static IP addresses because some client applications require configuration with a server's IP address rather than a server's name. You'll also need the network mask (see Chapters 3 and 11 for details about TCP/IP addressing), the IP address of the server's primary gateway (in other words, the default gateway), and the IP address of the new server's TCP/IP domain name server.
- *What kind of video card is installed in the server?* The Linux setup process attempts to detect the video card and will install the correct driver if possible. Otherwise, it will prompt you to choose the type of video card from a list. Either way, you should know what kind of video card your server contains.
- *What kind of monitor is attached to the new server?* As yet, an operating system cannot automatically learn this information. You'll need to supply the monitor manufacturer and model number to the Linux setup program.
- *What is the administrative user's password?* Choose a difficult-to-guess password for the Linux administrator account. Chapter 15 provides advice on choosing good passwords.

- *How can I remember all of this information?* Once you have answered these questions, you should create a server installation form and keep the form with you during installation. Appendix C offers an example of such a form.

This list highlights only the most significant installation options. In addition, you should be prepared to identify your keyboard and mouse type, choose a time zone, and specify an administrator account ID.

Although this example involves a standalone Linux system, Linux very peacefully coexists with other operating systems on your primary hard drive. (Read more about multiboot systems at the Linux installation **HOWTO** site:

www.linuxhq.com/ldp/howto/Installation-HOWTO/index.html.

For this installation, you'll need the following:

- A clean PC (one without any operating system installed) that satisfies the Linux hardware requirements detailed previously
- The distribution media for Red Hat Professional Server version 7.0 (CD-ROM and floppy disk)
- Two hours of available time, most of it for the installation process to copy files from the installation CD-ROM to your computer's hard drive

The Installation Process

The package containing Red Hat Professional Server version 7.0 includes a manual, 10 CD-ROMs, and a bootable floppy disk labeled "Boot Diskette." Place the first Red Hat CD-ROM near your computer. You won't need the floppy disk if your computer's BIOS configuration allows you to boot from the CD-ROM. Turn off your computer before beginning the installation. As you work your way through the installation screens, keep in mind that you can use the mouse to click the desired button. Use the Tab key to select actions, and press Enter to proceed from one screen to the next.

To install Red Hat Linux version 7.0:

1. Turn on your computer and insert the first CD-ROM disk into your computer's CD-ROM drive.
2. After reading the CD-ROM, the computer displays the Welcome to Red Hat screen, as shown in Figure 10-4.

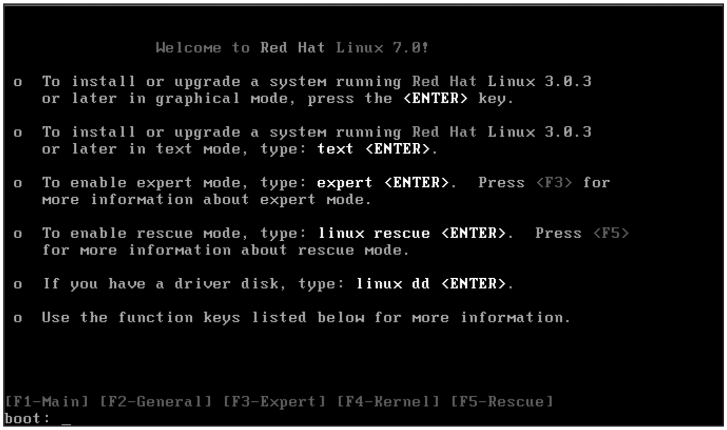

Figure 10-4 Welcome to Red Hat Linux screen

3. Press **Enter**. The Language Selection screen appears. Choose the language you'll use by clicking your choice with the left mouse button. Click **Next**.
4. The Keyboard Configuration screen appears. Use the scroll bars at the right of each section to locate your keyboard model and layout. Click your choice of keyboard model and layout. The highlighted selections should be fine for most English-speaking users with standard keyboards. Unless you know you need to disable "dead keys" do nothing in the "Dead Keys" section of this screen. Click in the Test your selection here text box and type a few characters. Click **Next**.
5. The Mouse Configuration screen appears. Red Hat Linux attempts to identify your mouse and highlights its choice for you. Choose your mouse by clicking the appropriate selection if you know your mouse is different from the choice highlighted. Expand and collapse mouse groups by clicking the **+/-** symbol at the left of the mouse list box. Serial port mouse users must also choose the COM port to which the mouse is connected. Click the appropriate selection to choose your mouse port. Click the **Emulate 3 buttons** check box if your mouse has only two buttons. Click **Next**.
6. The Welcome to Red Hat Linux screen appears. Read the instructions in the left panel, using the scroll bar to read all of the text. Click **Next**.
7. The Install Options screen appears. Click the **Server System** icon. Click **Next**.
8. The Initialize dialog box appears. Click **Initialize** to continue.
9. The Automatic Partitioning screen appears. Select automatic partitioning by clicking the **Automatically partition and remove data** diamond. Click **Next**.

Automatic partitioning will destroy *all* the data on your hard drive. Ensure that this is what you want to do before continuing.

10. The Network Configuration screen appears as shown in Figure 10-5. Click in the IP Address text box and type the IP address you chose for this server. Press **Tab** to continue to the Netmask text box and enter the correct network mask for this server. Press **Tab** to move to the Network text box; the network address in the Network text box should be correct. If it is not correct, change it now, and then press **Tab** to continue to the Broadcast text box, which should contain the correct broadcast IP address for this server. (If it is not correct, change it now.) Click in the Hostname text box and type the name of this server. Press **Tab** to continue to the Gateway text box and enter the IP address of this server's primary gateway. Press **Tab** to continue to the Primary DNS text box and type the IP address of this server's primary domain name server. Enter the IP addresses of the secondary and ternary (third) domain name servers if you know them. Click **Next**.

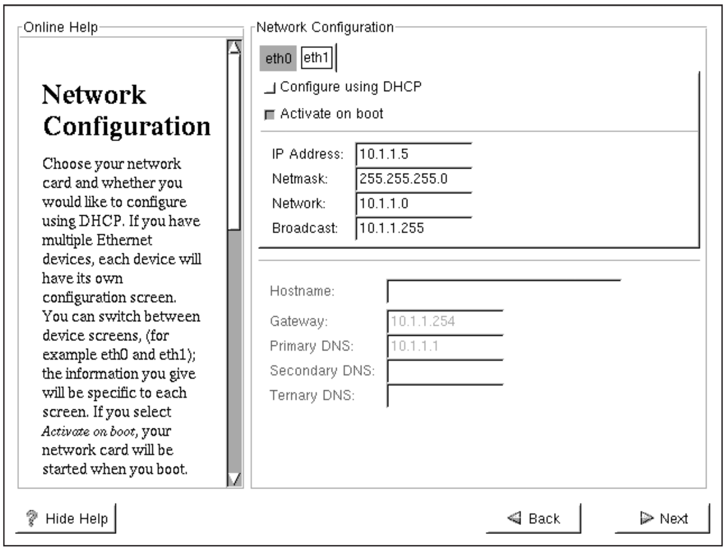

Figure 10-5 Network Configuration screen

11. The Time Zone Selection screen appears. You may select the appropriate time zone for this server in one of two ways: click the visual map of the world, or locate in the list below the map the country and city closest to the server's location and select it by clicking it. Using either method of choosing the time zone, a red "x" will appear on the map indicating the geographical location of the city you chose. Click the **System clock uses UTC** check box if your server's hardware clock is not set to the local time. Click the **UTC Offset** tab at the top of the Time Zone Selection box. Locate your time zone in the list and then click the appropriate selection. Click **Next**.

10

12. The Account Configuration screen appears. Click in the Root Password text box and type a password for the server's administrator (or root user). Press **Tab** and retype the root password in the Confirm text box. Note that the Next button remains gray (that is, inactive) until a valid root password and confirmation are entered. Do not enter a new account name. Click **Next**.
13. The Selecting Package Groups screen appears. The Web Server check box is selected by default. Make sure the Select individual packages check box is not selected. Click **Next**.
14. The About to Install screen appears. Click **Next**.

Caution

This is your last chance to stop before the installation process erases *all* the data on your hard drive.

15. The Installing Packages screen appears with a status table and progress bars, indicating the installation progress of individual packages and the overall installation. This is the most time-consuming step in the installation and will take more than an hour. The Next button will change from gray to black and will become active when package installation is complete. Click **Next**.
16. The Boot Disk Creation screen appears. Insert a blank, formatted floppy disk in the server floppy disk drive, and click **Next**.
17. Log in to your new Linux server using the user name **root**. The system presents you with the login prompt. Use the password you entered in Step 12.

Configuring Linux for Network Administration

A Linux server is little more than a powerful workstation when it has no user accounts. This section introduces you to the setup process for Linux system administration. You'll learn:

- The basics of adding users and groups
- The basics of modifying file access permissions

This section introduces two commands: `groupadd` and `useradd`. Both are documented with their own manual pages. Their names imply their function: `groupadd` enables you to add a new group to the system, and `useradd` enables you to add a new user to the system.

Like Windows 2000 and NetWare, Linux requires the use of user names and passwords to connect clients to the network. Also like these operating systems, it assigns access rights to groups, and allows users to be members of multiple groups. For example, the Linux group named *mail* can access the electronic mail programs and electronic mail files. This section assumes that you are logged in to a Linux system as the administrative

user ("*root*") and that your system has presented you with a command prompt. You'll press Enter after typing each command to allow Linux to carry out the operation. You may want to reread Table 8-2, which covers security for groups, because that information is relevant here.

Adding Groups and Users

You use the `groupadd` command to add a new group ID to a Linux system. It does not assign users to the new group, but rather makes a new group name available for use. Linux assigns a unique identification number to each group. Note that creating a new group does not automatically assign access rights to that group; you'll learn how to accomplish that task later in this section. You'll use the same school groups and access rights given in Table 8-2 for this section. Creating a new Linux group for instructors is simply a matter of typing the necessary commands. Note that the commands display no information if they successfully complete the operation.

To add group IDs to your Linux system:

1. Type **`groupadd instructors`**, and then press **Enter** at the command prompt. The group *instructors* is added.
2. Type **`groupadd students`**, and then press **Enter**. The group *students* is added.
3. Type **`groupadd administrators`**, and then press **Enter**. The group *administrators* is added.

You use the `useradd` command to add a new user ID to a Linux system. It creates a new user ID and assigns that user ID to one or more groups. In this example, you'll create a new user, Thomas, and assign that user to the group *instructors*. The new user should be a member of the general users group as well as the group *instructors*. You must use two options when typing the `useradd` command: the `–g` option specifies the initial (or primary) group for the user, and the –*G* option specifies the additional groups to which the new user will belong (*instructors,* in this case). Note that `useradd` does not assign a password for the new user ID, so you'll use the `passwd` command to assign a password for *thomas*.

> Linux passwords are case-sensitive. You may use any of the characters on the keyboard in your password.

You can use the `passwd` command in one of two ways: while logged onto the system as the administrative user, root, to change another user's password, or while logged onto the system as a normal user to change your own password. As you type the password, notice that the characters do not appear on the screen. This security precaution prevents people from peering over your shoulder and seeing your password as you type it. Read the `passwd` manual page `(man passwd)` to learn more about this command.

To add a new user and assign the user a password:

1. Type **`useradd -g users -G instructors thomas`**, and then press **Enter** to add a new user account named *thomas*.
2. Type **`passwd thomas`**, and then press **Enter**. Linux prompts you to type the new password. After you type the password and press **Enter**, Linux prompts you to retype your password. Enter the same password again; this confirmation helps ensure that you type your new password accurately.

Now that you've added a new group and a new user to the system, you may restrict access to resources owned by the user *thomas* or the group *instructors*.

Changing File Access Permissions

Linux restricts access to resources by comparing user and group IDs with the owner and group membership of files. Every file and directory on a Linux system is owned by exactly one user and is a member of exactly one group. You may assign access permissions for the file's owner, the file's group, and everyone else. Linux assigns new files and directories to the creator's primary group (*users,* in this example). The example from Table 8-1 shows that the directory *PROGRAMS* contains instructional programs. You want to allow teachers to place new programs in *PROGRAMS*. Students should be able to run the programs, but not to add new ones or to delete them.

To create a directory and assign it to a group:

1. To log off your Linux system, type **exit**, and then press **Enter**.
2. To log back onto your system as user *thomas,* enter **thomas** at the login prompt.
3. Enter the password you assigned for *thomas*.
4. You see a command window and a command prompt. To create the new directory, type the command **`mkdir PROGRAMS`**, and then press **Enter**.
5. List the file with **`ls -l`**. Notice that the directory belongs to the group *users.*
6. Type the command **`chgrp instructors PROGRAMS`** (remember to press **Enter**) to assign *PROGRAMS* to the group *instructors*.
7. List the files again with **`ls –l`** and notice that the file is now assigned to the group *instructors*.

Now that you've created the directory *PROGRAMS* and assigned it to the group *instructors*, you must limit access to the files. You use the `chmod` command to change the access permissions of files and directories. Read about `chmod` in its manual page (using the command `man chmod`). Your goal is to enable members of the group *instructors* to create new files in and delete files from *PROGRAMS*, and to limit access to all others (specifically, members of the group *students*). To accomplish this task, you must add write

permission to PROGRAMS for the group (in this case "instructors") and remove write permission for all others.

To change the access permissions for the PROGRAMS directory:

1. Type **`chmod g+w PROGRAMS`** to add write access for the *instructors* group to *PROGRAMS*, and then press **Enter**.
2. Type **`chmod o-rw PROGRAMS`** to remove read and write access by others to *PROGRAMS*, and then press **Enter**.
3. Type **`ls -l`** to view the access permissions assigned to *PROGRAMS*. You should see a line for *PROGRAMS* that includes permissions of *drwxrwxr-x*.

INTERNETWORKING WITH OTHER NETWORK OPERATING SYSTEMS

People have modified the UNIX system over the years to work with other network operating systems and protocols besides TCP/IP. Programmers and network administrators alike have added functionality to the system because they find it so productive. Their changes include the addition of Windows networking, NetWare networking, IBM mainframe terminal emulation, and Windows programming tools. Examples of these tools are described below.

- *Samba*—The UNIX-based server message block (SMB) and common Internet file system (CIFS) package. This application provides everything needed to make your UNIX system a fully featured Windows file and printer-sharing server.
- *IPX/SPX*—The original Novell networking protocol. It is implemented as a native UNIX protocol in many UNIX versions. Proprietary and open source software solutions, such as Caldera's NetWare for Linux, exist to turn your UNIX server into a NetWare server by including full NDS support.
- *AppleTalk*—The Apple Macintosh network protocol. Many UNIX system vendors support this protocol and all application programs needed to implement Macintosh file and print servers.
- *X3270*—An X Window–based 3270 terminal emulator for accessing your mainframe over a TCP/IP connection. This standard application is included with the X Window system for UNIX.
- *WINE*—An open source application that implements a Win32 programming subsystem for UNIX including support for running Windows programs (such as Word). This ongoing project is slowly improving with age as more people begin to use it and contribute changes to the source code.
- *VMWare*—A commercial application that emulates a complete Intel-based computer. VMWare uses special Intel CPU instructions to perform its emulation. This hardware support for emulation enables VMWare virtual computers

to run almost as fast as the actual computers themselves. Unlike WINE, VMWare emulates the computer hardware (rather than the programming environment) and can run any Intel-based operating system.

- Dozens and dozens of command-line utilities that enable you to access the contents of files generated on other systems

CHAPTER SUMMARY

- The UNIX system is a stable, robust network operating system. It forms the basis of much of the Internet. You must be familiar with the operation of UNIX so as to set up and maintain most local and wide area networks. Despite the preponderance of proprietary implementations of UNIX systems, the differences between the various versions are relatively minor.
- UNIX was born at AT&T's Bell Laboratories, when a few programmers grew dissatisfied with the programming environments available in the late 1960s. Ken Thompson and Dennis Ritchie were the original authors of the system.
- Currently, Caldera International and Tarantella, Inc. own the rights to the UNIX source code. The Open Group, a nonprofit trade association, owns the UNIX trademark.
- Sun Microsystems, IBM, and Hewlett-Packard sell the three most popular UNIX-based workstations. These products are based on these companies' proprietary implementations of UNIX, which conform to most UNIX standards.
- Recently, nonproprietary implementations of UNIX-like systems have become popular. Often referred to as open source software or freely distributable software, this category includes the UNIX-like systems FreeBSD, GNU, and Linux.
- The key difference between freely distributable UNIX and proprietary implementations is that the copyright on freely distributable implementations requires that anyone purchasing an open source version of UNIX receive access to the source code.
- UNIX systems make great Internet servers. In fact, the leading Internet Web server is an open source software project called Apache. The original Web tools—including browsers and servers—were developed on UNIX-based systems. UNIX systems underlay the original ARPAnet and Internet services such as FTP, Telnet, gopher, HTTP, and POP. These services are standard with current implementations of UNIX.
- One characteristic of all UNIX systems is a user-definable command interpreter. Its development arose in part because the command line was the primary interface with the system.
- Other characteristics of UNIX systems are as follows: the ability to support multiple, simultaneous users; hierarchical file systems with demountable volumes; a consistent interface for files, devices, and interprocess input/output; hundreds of subsystems and dozens of programming languages; program source code portability between different implementations of the system; and user-definable windowing systems.

- Minimum hardware requirements for a Linux server include an Intel-compatible x86 processor, 32 MB RAM, 500 MB of hard disk space, a network interface card compatible with the rest of your network, a CD-ROM drive, and a floppy disk drive.
- The UNIX system was among the first operating systems to include a hierarchical file system. This approach led to better-organized data and subsystems.
- Each UNIX process represents an instance of a running executable program in core memory (RAM). The UNIX kernel allocates separate resources (such as buffer space, stack space, and open file pointers) to each process as it is created. The kernel manages access to these resources.
- You can liken UNIX commands to ordinary sentences. The things you want the system to operate on are the nouns—often files. Options to the commands are the adjectives and the adverbs.
- Most UNIX commands are lowercase alphabetic characters. To specify an option, you usually type a dash ("`-`") followed by a letter. The letter is often (but not always) a mnemonic abbreviation for the option (such as *`-l`* for a long file listing).
- Command names are usually acronyms or abbreviations. Consult the command's manual (man) page when you encounter a command name that makes no sense to you. The synopsis usually indicates the origin of the command name.

- The UNIX *`ls`* command is the most frequently used. When you use *`ls`* with the *`-l`* option, it allows you to learn everything about a file except its contents. *`ls -l`* reports the filename, the file size, the date and time that the file was created, the date and time that it was last accessed, the date and time that it was last modified, the number of "aliases" or links to the file, the user who owns the file, the group to which the file belongs, and the access rights for the owner, the group, and all others.
- The system uses information nodes (i-nodes) to store everything other than the actual contents of files. I-nodes also contain pointers to file contents on the disk.
- Linux distributions are binary compatible. They differ mainly in the methods they use for installing additional software packages. The Red Hat Linux distribution is one of the most popular.
- Use the **`useradd`** command to add new users to your Linux system.
- Use the **`groupadd`** command to add new groups to your Linux system.
- The **`chgrp`** command assigns a file to a group.
- The **`chmod`** command changes file access permissions.
- UNIX systems quite competently interoperate with other network operating systems. You can use them to share files with Windows-based computers and NetWare-based computers, for example. In addition, you can use a UNIX-based computer to access mainframe sessions. You can even run Windows software on UNIX systems with the proper emulation package installed.

Key Terms

AIX — IBM's proprietary implementation of the UNIX system.

Apache — A popular open source software Web server application often used on Linux Internet servers.

BSD (Berkeley Software Distribution) — A UNIX distribution that originated at the University of California at Berkeley. The BSD suffix differentiates these distributions from AT&T distributions. No longer being developed at Berkeley, the last public release of BSD UNIX was version 4.4.

command interpreter — A (usually text-based) program that accepts and executes system programs and applications on behalf of users. Often it includes the ability to execute a series of instructions that are stored in a file.

doskey — A command used on MS-DOS and Windows systems that enables the user to recall (using the keyboard's arrow keys) and edit previously entered commands.

flavor — Term used to refer to the different implementations of a particular UNIX-like system. For example, the different flavors of Linux include Red Hat, Caldera, and Mandrake.

FreeBSD — An open source software implementation of the Berkeley Software Distribution version of the UNIX system.

freely distributable software — A term used to describe software with a very liberal copyright. Often associated with open source software.

General Public License — The copyright that applies to freely distributable versions of UNIX and specifies that the source code must be made available to anyone receiving the system.

globbing — A form of filename substitution, similar to the use of wildcards in Windows and DOS.

GNU — The name given to the free software project to implement a complete source code implementation of UNIX, the collection of UNIX-inspired utilities and tools that are included with Linux distributions and other free software UNIX systems. The acronym within an acronym stands for "GNUs Not UNIX."

hierarchical file system — The organization of files and directories (or folders) on a disk partition in which directories may contain files and other directories. When displayed graphically, this organization resembles a tree-like structure.

HOWTO — A series of brief, highly focused documents giving Linux system details. The people responsible for the Linux Documentation Project centrally coordinate the HOWTO papers (see *www.linuxhq.com/ldp/howto/HOWTO-INDEX/howtos.html*).

HP-UX — Hewlett-Packard's proprietary implementation of the UNIX system.

Hurd — The kernel in the GNU operating system. While many UNIX and Linux systems include GNU utilities such as the EMACS editor or the GNU C compiler, the Hurd is the only operating system kernel that can currently be called a GNU kernel.

i-node — A UNIX file system information storage area that holds all details about a file. This information includes the size, access rights, date and time of creation, and a pointer to the actual contents of the file.

kernel — The core of a UNIX system. This part of the operating system is loaded and run when you turn on your computer. It mediates between user programs and the computer hardware.

kernel modules — Portions of the Linux kernel that you can load and unload to add or remove functionality on a running Linux system.

Linux — A freely distributable implementation of the UNIX system. Finnish computer scientist Linus Torvalds originally developed it.

manual pages — UNIX online documentation. This documentation describes the use of the commands and the programming interface to the UNIX system.

NFS — Network File System. A client/server application that allows you to view, store and update files on a remote computer as though they were on your own computer. Can be used to install Linux.

open source software — Term used to describe software that is distributed without any restriction and whose source code is freely available. See also *freely distributable*.

Orange Book — A rigorous security specification for computer operating systems published in 1985 by the U.S. Department of Defense.

pipe — The facility in a UNIX system that enables you to combine commands to form new commands. It is one of the most powerful facilities of the UNIX system.

pipeline — A series of two or more UNIX commands connected together with pipe symbols.

proprietary UNIX — Any implementation of UNIX for which the source code is either unavailable or available only by purchasing a licensed copy from Caldera International and Tarantella, Inc. (costing as much as millions of dollars).

real-time — The term used to describe an operating system that includes at least one of the following two characteristics: the ability to respond to external events (for example, a change in temperature), and an ability to respond to those events deterministically—with predictable response time (for example, turning on a heating element within three microseconds).

Samba — An open source software package that provides complete Windows 2000-style file and printer sharing facility.

shell — Another term for command interpreter.

Solaris — Sun Microsystems' proprietary implementation of the UNIX system.

source code — Computer instructions written in a programming language that is readable by humans. Source code must be translated into a form that is executable by the machine, typically called binary code (for the sequence of zeros and ones) or target code.

System V — The proprietary version of UNIX, originally developed at AT&T Bell Labs, currently distributed by Caldera International and Tarantella, Inc.

time-sharing system — A computing system to which users must attach directly so as to use the shared resources of the computer.

Review Questions

1. In what year did work begin on the UNIX system?
   a. 1990
   b. 1987
   c. 1975
   d. 1969
2. Which two of the following are open source software implementations of UNIX-like systems?
   a. Solaris
   b. IRIX
   c. Linux
   d. FreeBSD
   e. HP-UX
3. Which of the following services might not be appropriate for a UNIX server?
   a. Internet services
   b. file and print services
   c. image database services
   d. music on hold service
   e. DHCP service
4. It is appropriate to use UNIX systems for network firewalls. True or False?
5. What is the primary method for interacting with the UNIX system?
   a. speaking to it through a microphone
   b. moving a mouse and clicking icons
   c. typing commands at a command prompt
   d. typing commands in a dialog box
6. Which of the following is a characteristic common to all UNIX systems?
   a. the same font for all windows
   b. availability in multiple foreign languages
   c. the ability to start processes in the background
   d. file systems in which directories cannot contain other directories
   e. the availability of Windows emulation programs

7. Which character is used to separate directory names on UNIX systems?
   a. backslash
   b. colon
   c. comma
   d. period
   e. forward slash
8. Hardware requirements for UNIX servers are roughly equivalent to those of Windows 2000 or NetWare servers. True or False?
9. Which of the following are stored in a file's i-node?
   a. access rights
   b. the filename
   c. the first 16 bytes of the file
   d. the time and date that the file was last printed
10. Which letter does the `ls` command use to identify a UNIX symbolic link (much like a Windows 2000 shortcut) in a detailed file listing?
    a. L
    b. S
    c. l
    d. y
    e. s
11. Which two of the following items are you required to know when installing a Red Hat Linux server?
    a. your Internet service provider's name
    b. your printer brand and model
    c. the number of buttons on your mouse
    d. the server's IP address
12. The administrative user "root" is subject to which of the following access restrictions, just like any other UNIX system user?
    a. read access
    b. create access
    c. write access
    d. execute access
    e. none of the above

13. Which open source software application enables UNIX systems to participate in SMB file sharing on a network?
    a. Tango
    b. Samba
    c. Jitterbug
    d. Waltz
14. Which three of the following tools might enable you to run Windows programs from your Linux system?
    a. Samba
    b. X3270
    c. WINE
    d. Windows 2000
15. Which command would you use to create a new directory on a UNIX system?
    a. `make`
    b. `makedir`
    c. `mkdir`
    d. `md`
16. Which command would you use to remove a directory on a UNIX system?
    a. `mkdir`
    b. `rmdir`
    c. `deldir`
    d. `rd`
    e. `removedir`
17. Which option would you use to tell the `ls` command to display all files in the current directory?
    a. `—all`
    b. `-listall`
    c. `/a`
    d. `—a`
    e. `/all`
18. What does `grep` stand for?
    a. globbing relocation expert processor
    b. general replication executable pointer
    c. generic regular expression pointer
    d. general regular expression parser
    e. none of the above

19. Which UNIX command might you use to display the last 10 lines of a file?
    a. `grep`
    b. `exit`
    c. `tail`
    d. `who`
    e. `list`
20. Under what circumstances would file globbing be useful?
21. If a UNIX command is like a sentence, which part of the command is like the sentence's verb?
    a. the command name
    b. the command options
    c. the Enter key
    d. the filename associated with the command
22. Which part of a UNIX command is like the noun in a sentence?
    a. the command name
    b. the command options
    c. the Enter key
    d. the filename associated with the command
23. How would you learn about a command for which you know the name, but not the purpose?
    a. Use the `help` command.
    b. Use the `manual` command.
    c. Use the `man` command.
    d. Use the `apropos` command.
24. How would you learn about a command for which you know the purpose, but not the exact name?
    a. Use the `help` command.
    b. Use the `manual` command.
    c. Use the `man` command.
    d. Use the `apropos` command.
25. What is the full command you would use to revoke write permission for everyone other than yourself for the file named *sent-mail?*
    a. `chown root sent-mail`
    b. `chmod —write sent-mail`
    c. `chmod og-w sent-mail`
    d. `chgrp system sent-mail`

26. What is the full command you would use to create the directory named *GRADES*?
    a. `cp new GRADES`
    b. `makedir GRADES`
    c. `mkdir GRADES`
    d. `make dir GRADES`
27. What is a Linux kernel module?
    a. a shared Linux library
    b. the same thing as a Linux process
    c. a small part of the Linux source code
    d. part of the Linux kernel that you can load and unload to add or remove functionality on a running Linux system
28. A UNIX thread is:
    a. the same thing as a UNIX process
    b. sometimes called a lightweight process
    c. what holds the UNIX kernel together
    d. the portion of RAM that holds the process information
29. Which two of the following choices describes how the open source application WINE differs from the commercial application VMWare?
    a. They are not different, they do the same thing.
    b. WINE implements the Win32 programming interface, and VMWare emulates a complete personal computer.
    c. WINE emulates a complete personal computer, and VMWare implements the Win32 programming interface.
    d. VMWare enables you to run any Intel-based operating system, but WINE runs only Windows programs.
    e. VMWare runs only Windows programs, but WINE enables you to run any Intel-based operating system.
30. What is the UNIX `telnet` command used for?
    a. to start the networking system
    b. to stop the networking system
    c. to access the Ethernet interface
    d. to begin a virtual terminal connection to a remote server
    e. to initiate a network broadcast

Hands-on Projects

Project 10-1

In this exercise, you'll learn about some of the basic differences between the Windows (or DOS) command line and the UNIX command line. To complete this project, you'll need a computer system running Red Hat Linux Version 7.0. If you have not already done so, install Linux by following the steps in this chapter, using a computer that meets at least the minimum hardware requirements, also described in this chapter.

1. Turn on your computer and wait for it to start Linux.
2. Type your user name and password at the login and password prompts.
3. The system presents you with a command-line window (usually called a terminal window) and a command prompt. Type a slash character (`/`), then press **Enter**. What happened? The command interpreter was awaiting a command, and you typed a directory name (/).
4. Type a backslash character (\), then press **Enter**. What happened? The backslash character tells the system not to process the command, but to wait for the rest of the command. You can use this feature to enter very long commands on multiple lines and still maintain the readability when you print a copy of the command.
5. Press **Enter** again. What happened this time? The backslash you typed in Step 4 instructed the system to wait for more input, but the rest of your input was empty. The system did nothing and displayed the command prompt, awaiting your next command.
6. Press **Ctrl+D**. What happened? Ctrl+D is a special control sequence, signaling to the command interpreter that it should end current processing. As it was sent to your login command interpreter, you signaled to the system that you were logging out.

Project 10-2

In this exercise, you'll get more exposure to the UNIX `ls` command. Keep the "verbs," "nouns," and rules in mind when trying this exercise, and remember to press Enter after each command. You'll need a running Linux system for this exercise—that is, a computer that meets the minimum hardware requirements with a copy of Linux installed.

1. Turn on your computer and wait for it to start Linux.
2. Log on with your user name and password.
3. Type **`ls`** at the command-line prompt. What do you see? Don't be alarmed. UNIX designers subscribe to the notion that "no news is good news." The fact that running `ls` generated no output simply means that it found no files to list. This response is quite normal for brand new user accounts.

4. Add the adjective "all" to the verb as follows: **`ls –a`**. In what way is this command different from that given in Step 3? Linux "hides" filenames beginning with a period, much as Windows Explorer can hide files with certain extensions.
5. Read the manual page for *ls* `(man ls)`. How does `ls` sort the output by default? How can you instruct `ls` to sort the output by the files' timestamps instead?
6. Tell the system you want to see the details of all files in the current directory `(ls –la)`. How is this command different from the listing performed in Step 4?
7. Produce a detailed listing of all files in the directory `/usr/bin`, sorting the listing by file timestamp in reverse date order `(ls -alrt /usr/bin)`. Is there a different way to specify the options in this command? Many UNIX commands accept both short and long options. You invoke `ls` with long options by typing two hyphen characters followed by the descriptive option name. The `ls` manual page documents the long and the short options.

Project 10-3

This project introduces the notion of connected commands. The connection method you'll use is a pipe, the mechanism that directs the output and input of commands. UNIX pipes are used to combine the output of one command with the input of another command; they represent one of the most powerful features of the UNIX system, and a feature that has been mimicked by many other computing systems, including Windows 2000. As explained in this chapter, the symbol used for a pipe in the UNIX shell is "`|`", the vertical bar, usually located on computer keyboards above the backslash. You type a pipe symbol by pressing SHIFT and backslash. A UNIX pipeline is simply two or more commands on one command line with a "`|`" between them. You'll use two commands in this project: `who` and `grep`. The `who` command displays the users who are currently logged onto the system. Running `who` on a busy server might show as many as 100 users logged on. The UNIX command `grep` is used to search for strings of characters in files. As in the previous projects, you'll need a running Linux system to complete this exercise. You must be logged into your lab system at a shell prompt.

1. Type **`who`**, then press **Enter**. You should see a list of all logged-on users (possibly just one).
2. Press **Alt+F2** to switch to a new terminal window. At the login prompt, type **thomas** and press **Enter**. At the password prompt, type the password you chose in the exercise in "Adding Users and Groups." Press **Alt+F1** to switch back to the first terminal window.

Note

Red Hat Linux systems are configured with seven "virtual" terminal windows, all of which are active at the same time. You access them by pressing **Alt+F1**, **Alt+F2**, **Alt+F3**, and so on. This feature offers you the possibility of starting up to seven simultaneous login sessions without using a graphical user interface.

3. Type **`who | grep root`**. You should see just one line displayed: the line representing the login information for the user *root*.

Project 10-4

Linux supports many different network services One of the most basic network services is remote host command-line access. Telnet is the Internet remote command-line service. In this exercise, you'll access a Linux server with a Windows telnet client. For this exercise, make sure that the server you built is running Linux. You also need to be logged into a Windows 2000 workstation.

1. To start the Windows Telnet client click **Start** and click **Run**. The Run dialog box opens.
2. Type **Telnet** in the text box and click **OK**. The Telnet window opens.
3. Type **open** followed by the IP address of the Linux server (**10.1.1.5** in this example) and press **Enter**. Figure 10-6 shows an example Windows Telnet session. The Telnet session starts, and you are presented with a login prompt.

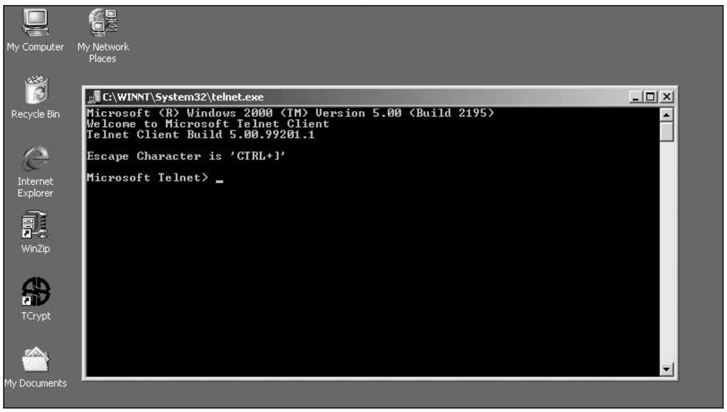

Figure 10-6 Windows Telnet Session

4. Type the user name **thomas** at the login prompt and press **Enter**. A password prompt appears.
5. Enter the password you chose when you added the user *thomas* above. You see a shell prompt. You are now logged into the remote Linux server via Telnet. The Telnet window on the Windows workstation is connected to the Linux server just as if you were using the keyboard and monitor directly attached to the Linux server. Type **`ls –l /`** and then press **Enter**. Figure 10-7 shows the output of the *ls –l /* command.

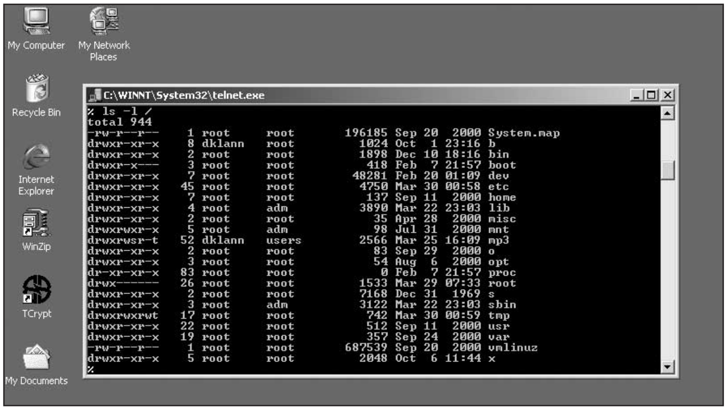

Figure 10-7 Output of `IS-1/`command in a Telnet session

6. Type **exit**, and press **Enter** to end your Telnet session.

Case Projects

Case Project

1. Rick Gomez, the director of product development for EarTech, has asked you to investigate setting up a test network for the company's product development team. The network will support the company's new "EarRadio 2010" FM radio cochlear implant. The network should be connected to the company LAN, but only through one system and only for purposes of Internet access. All of the usual services (such as file sharing, printing, and backup) will be handled on the local LAN. Rick suggests that you plan for a maximum of 30 users, 12 of whom might use the network at any given time. After investigating the situation, you learn that the test network will include several kinds of workstations: Windows PCs, Hewlett-Packard UNIX workstations, and Silicon Graphics UNIX workstations, plus two networked read-only memory (ROM) programming devices (these look to the users very much like network printers). Draw a network diagram that includes the following items:
   - The connection to the existing company LAN
   - The gateway between the test network and the company LAN
   - Eight workstations
   - The file server
   - The print server
   - The printer
   - The ROM programming devices

You also learn that the company has an unused Intel-based PC that has two network interface cards installed, and you have received permission to use this computer as the gateway between the test network and the company LAN. Why might you choose to run Linux on the gateway? How would you learn whether the NICs in the gateway are supported by the Linux operating system?

2. After mapping the network, you learn that the users of both workstation types need to share some common files. Describe how you might set up a single file server that could handle the requirements of both Windows and UNIX network file systems. Which facilities of a UNIX server might help to share files with workstations running both of these operating systems?
3. Knowing that UNIX systems provide superior print spooling service as well as file sharing, you decide to streamline the network even more. Describe how you might eliminate a separate print server from the network. Do you think the UNIX file server could be used as a print server as well? Why or why not?
4. After you install Linux on the gateway you want to ensure that IP packets will pass from the workstations on the test network to the Internet. How would you determine whether the gateway was properly set up to route packets between the two networks?